École Polytechnique - LIX - GeoViC

Deep 3D Point Cloud Denoising

Diego Gomez

May - July 2022

Contents

1 Introduction

Point cloud structures are of raising importance in today's increasingly virtual world. For instance point clouds come into play in famous technologies such as lidar, now present in every recent iPhone. Due to their simple nature, but loyal and meaningful representation these structures are also of interest in research topics, such as object recognition [17]. Their use assumes that the collected points are corruption free. However, this assumption does not generally hold with current technology. Denoising techniques for lidar scans have been around for almost a decade [24]. Indeed, lasers acquisitions or reconstruction algorithms often result in point clouds corrupted with noise and outliers.

The cloud denoising problem has been thoroughly investigated via various methods. The survey [6] provides a recent overview of this field. Classical methods include statistically-based [19] ones or even techniques that rely on Partial Differential Equations (PDE's) [2]. Since the pioneering presentation of PointCleanNet [16] the research on data driven methods has risen [23]. The deep learning methods include some based on gradient estimation [12], some that use Graph Neural Networks [13] and others that take unsupervised approaches [7]. Early statistical methods dealt mainly with the cloud "cleaning" problem by focusing on outlier detection. The standard procedure of first removing outliers and then denoising has since remained. Some techniques such as PointCleanNet [16] describe methods to do both in a single pipeline. However, no method, to the author's knowledge manages to perform both outlier and noise removal in a single pass (for instance PointCleanNet requires two separate networks).

Inspired by the success that the learning based methods have achieved we focus in developing a new deep learning method to tackle the cloud denoising problem. Our goal is to further our understanding of the current methods.

1.1 Problem Statement

We first provide a formalization of the problem at hand. It is inspired from the one presented in [16].

We can define the task of denoising as in signal processing. Let $\mathcal{S} \subset \mathbb{R}^3$ be a 2-manifold surface of $\mathbb{R}^3$ and let $\mathbb{P}_{\mathcal{S}}^{C} = \{p \mid p \in \mathcal{S}\}$ be a discrete sampling of its surface. A *noisy* version of the resulting point cloud designates the point cloud obtained after moving each of its points by a random displacement $\mathbf{n} \sim \mathcal{N}$, where $\mathcal{N}$ designates the underlying noise model:

$$\mathbb{P}_{\mathcal{S}}^{N} = \{\mathbf{p} + \mathbf{n} \mid \mathbf{p} \in \mathcal{S}, \mathbf{n} \sim \mathcal{N}\}.$$

The goal of a denoising model is to retrieve $\mathbb{P}_{\mathcal{S}}^{C}$ given $\mathbb{P}_{\mathcal{S}}^{N}$.

A harder version of the setting above consists in adding *outliers* to the noisy point cloud. Outliers are data points that do not belong to the underlying surface and that do not follow the noise distribution $\mathcal{N}$. The set of noisy points becomes:

$$\mathbb{P}_{\mathcal{S}}^{N} = \{\mathbf{p} + \mathbf{n} \mid \mathbf{p} \in \mathcal{S}, n \sim \mathcal{N}\} \cup \{\mathbf{o} \mid \mathbf{o} \in \mathcal{O}\}.$$

In the following we will delve mainly in the first setting.

1.2 Motivations

Despite their success, some of these methods (specially PointCleanNet) use a set of techniques that we wish to address:

- The use of iterative cleaning. That is applying the network several times. We believe that it is possible to have a method that directly outputs the desired result.

- The use of non intuitive loss terms.
- The use of a point per point prediction. We introduce a method that processes patches at a time.

2 Related work

In this report we follow a data driven approach for point cloud denoising. Let us begin by exploring some prior work that tackles this problem.

2.1 Non-learning denoising

The cloud denoising problem has been long studied [6]. The pre-deep-learning approaches come in a variety of different facets. In the beginning, the research focused on outlier removal in order to perform point cloud cleaning. For instance, the statistical methods [19] mainly classify outliers using, as the name indicates, statistical tools. Other techniques involve the projections of the noisy points to some estimated local surface, for instance [3] is based on the computation of the moving least squares (MLS). Moreover, many other techniques involving complex ideas have been developed throughout the years [2, 6], and it is not the goal of this report to do an extensive study on a subset of them. The goal of the author is to give a glimpse of the data-driven approaches, and mainly, to propose and describe a novel promising method.

2.2 PointCleanNet

The paper that introduced PointCleanNet [16] was the pioneer on the usage of deep learning for point cloud denoising. In their paper the authors introduced a two step method to eliminate outlier points, and displace inlier points to the underlying ground truth surface. It is important to note that PointCleanNet treats the cloud denoising problem as a local one. That is, the prediction of a point $p'_i \in \mathbb{P}$ only depends on a sub-patch $\mathbb{P}_i$ centered on p'_i. We are thus considering the second setup described in section 1.1. The architecture proceeds in two stages: first a network $g : \mathbb{R}^{(3\times N)} \mapsto \mathbb{R}^3$ is trained to classify outliers

$$\tilde{o}_i = g(\mathbb{P}_i)$$

where $\tilde{o}_i$ is the predicted probability that $p'_i \in \mathcal{O}$. The network g is then a network that inputs a patch centered at a given point, and outputs the predicted probability that this center point is an outlier. Define $\tilde{\mathcal{O}} := \{p'_i | \tilde{o}_i > 0.5\}$ the set of predicted outliers. We can thus obtain a cloud $\hat{\mathbb{P}} = \mathbb{P}/\tilde{\mathcal{O}}$. Afterwards a separate network $f : \mathbb{R}^{(3\times N')} \mapsto \mathbb{R}^3$ is trained to predict displacements on the noisy points to bring them closer to the underlying surface.

$$d_i = f(\hat{\mathbb{P}}_i)$$

In a similar fashion, f is then a function that takes a patch as input, and outputs the correction-displacement vector for the center point. We finally obtain, $\hat{p_i} = p'_i + d_i$ the denoised point. PointCleanNet follows then the above procedure, where f and g are trained separately. Both are based on the PCPNet [5] architecture. The complete PointCleanNet architecture can be seen in Fig. 1. This pioneering method [16] out-performs non-learning denoising approaches, while being on the contrary to the classical methods, parameter-free.

However, this method comes with some issues that we wish to correct. In the paper, the authors make use of a non intuitive second loss term to combat artefacts in the predictions (See appendix for example, Fig. 11). We believe that the reason that the aforementioned artefacts appear is that there is a lack of communications between the movements of each point. That all the

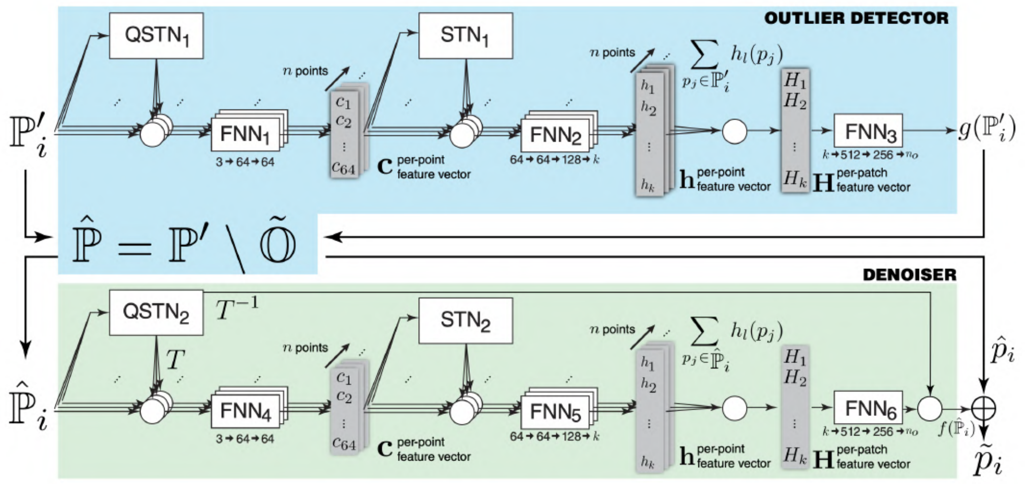


Figure 1: PointCleanNet architecture summarizing the above. For the details go to the PointCleanNet paper [16].

d_i's are independent. This stems from the fact that this method issues point per point predictions.

2.3 ScoreDenoise

A popular, learning based, denoising method is the ScoreDenoise architecture [12]. They claim that the noise can be modeled by $p * n$, where p is the underlying surface, n the noise distribution and $*$ the convolution operator. They define "score" to be,

$$\text{score}_i = \nabla_x (p * n)(x_i)$$

for x_i a noisy point with $i \in \{1, \ldots, N\}$. In the paper a network is trained in order to be able to compute this score for unobserved instances. Once one has this score, the method applies gradient ascent to move all points closer to the underlying surface. According to their results, this method outperforms other data-driven denoisers, including PointCleanNet. This however, is not a method that is well-aligned to our goals, since it does not solve one of the issues stated above: the use of iterative cleaning.

3 Network Design

In this section, we describe our proposed method to address the three issues raised in section 1.2, namely iterative cleaning, counter-intuitive losses and lack of point communication.

3.1 U-Net architecture

The U-Net architecture was firstly introduced in a paper [18] tackling the problem of biomedical image segmentation. Inspired by PCPNet [5], which re-purposes the PointNet architecture to regress local shape properties, we re-purpose a common semantic segmentation design for a regression task. This architecture gets its name from its characteristic “U” shape (see Fig. 2). The network down samples the features to a compact feature space, and up samples them again to obtain a prediction for every single input pixel/point.

In the KPConv paper [20] the authors adapt convolutions to three dimensional data. Using these convolutions as building blocks allows us to adapt the above U-net structure to be able to apply it to point clouds. Let us now describe the down and up sampling processes, since the adaptations to treat 3d data are not trivial.

Transition Down. To be specific, down-sampling is done via some kind of subsampling. In this report we use grid subsampling [11]. The cloud is projected into a grid, and one element of each cell is kept. Then the features are determined via the creation of a kNN graph, and applying a max pool operator on each sub-sampled point, i.e for a given sub-sampled point its sub-sampled feature is the max feature of its k nearest neighbors.

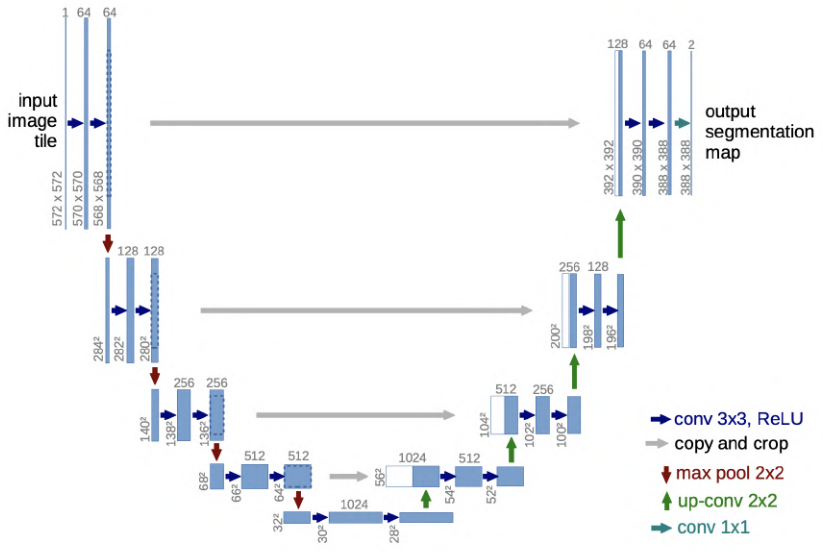


Figure 2: Original U-Net architecture, figure extracted from [18].

Transition Up. For up-sampling we use the U-Net design. All stages in the previously described encoder are paired with modules in the decoder. The decoder then uses the information on which points were down-sampled to up-sample them again. Then features are obtained via nearest neighbor interpolation as done in [14]. Finally, the interpolated features are aggregated with the skip-connections from the symmetric module on the encoder.

Output Head. In the case of semantic segmentation the head produces a prediction for every single point, with a dimensionality equal to the number of classes to predict. Each dimension represents a logit that encodes the probability of the point to belong to each class. In the case of the outlier detection problem this could be a scalar output for each point, where 1 would mean outlier and 0 inlier. For offset regression, we need to output a 3-dimensional vector, that represents the offset we need to add to the noisy point to "clean" it.

3.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) [4] are composed of two main components. A generative model G that fits and simulates the data distribution, and D a discriminate model that is trained to differentiate instances from the real data, and generated instances by G. These two components are trained in a minimax two-player game manner. Intuitively, this means that G is trying to maximize the error of D, since G wants to fool D; and D wants to be correct, thus trying to minimize its loss. Minimax games are also referred to as zero-sum games, games where there can only be one winner. The end goal is to obtain a model G that perfectly imitates the true data, in which case the discriminator D can only randomly guess the class of the input(i.e. it is maximally confused).

GANs have a solid use for denoising in the case of images [1]. They are in fact hugely relevant in the image domain. The success of such methods has reached a point where it is now a challenge[1] of itself, even for humans to differentiate GAN generated images from real ones [22]. Their use on 3D data is still not well established, even though there is significant ongoing research [9], some even with similar goals to ours like this up-sampling paper [10].

In this report we are mostly interested in the benefits that such a technique can provide to the training pipeline. For instance, in the paper "Globally and locally consistent image completion" [8], the authors use a discriminator component in order to train a network to complete missing holes in images. In the paper [8] the input of the completion model G are corrupted images, images with holes, and the output is the completed image (see appendix Fig. 10 for the structure). In the following report we attempt to translate this structure into the cloud denoising problem. More specifically, in this

[1]https://www.whichfaceisreal.com/index.php

case we would have a denoising network and the discriminator component. The goal is that the discriminator drives the denoising network to provide patches that have a greater resemblance to the underlying clean data.

4 Smoothness Study

In the following we study desirable properties from two of the presented architectures above: PointCleanNet and U-Net. For this section both networks have been trained on the PointCleanNet dataset [2] for the task of **outlier detection**[3]. However, we assume that the properties that are studied in the following section are proper to the given architectures, thus applicable for other tasks.

4.1 Statement of Test and Motivations

In this section, we verify experimentally whether the following assumption holds true: *the U-Net architecture outputs smoother predictions than PointCleanNet*. This statement is likely to be true because of the fundamental difference between both designs in terms of output: PointCleanNet *independently* predicts properties of each point, given a patch of points as input. U-Net uses and gives predictions for *all points* within the input patch. Smoothness is a desirable property of a "cleaning" architecture's output because nearby points are likely to require a similar properties. This is specially true in offset regression, similar points are likely to have similar offsets to be denoised. This property is leveraged in particular by ScoreDenoise [12], a state-of-the-art denoising network.

[2]`http://www.lix.polytechnique.fr/mrakotos/pointcleannet.htm`

[3]The study was done on the outlier detection problem mainly for time constraint issues, since I was given a pre-trained U-net model for this problem by my advisors Prof. Maks Ovsjanikov and Phd student Maxime Kirgo.

Our overall goal is to train a network that constructs an internal representation of the appearance of a clean patch. This would then potentially drive the regression of the offset that allows to bring neighboring noisy points to the closest point on the clean surface, or to predict the “outlierness” of a given point. Perhaps even manage both at the same time. Given that the U-Net architecture processes whole patches at a time, it has the means to organize the outputs if it deems it necessary. However, the PointCleanNet architecture lacks this ability, since it only has knowledge of where its center point will be translated to. Moreover, recall that in section 2.2 we have briefly discussed the fact that the PointCleanNet architecture needs a special unintuitive loss in order to yield regularly distributed point clouds. The U-Net architecture on the contrary has the structural equipment to establish a communication between all the points in the patch it processes. We hypothesize that this communication helps regularizing the predicted offsets without the need of special losses.

4.2 Results

4.2.1 Experiment Setup

The experiments were performed in an environment composed of 100 different center points (50 outlier points, 50 inlier points) in each shape available in our test set. Since our test set was composed of 31 shapes, we had a total of 3100 patches to study. The networks were used using pre-trained weights. The PointCleanNets’s ones were extracted from the main repository[4]. The ones for the U-net architecture were given to me by my advisors, from incoming work by Maxime Kirgo and Maks Ovsjanikov.

[4]`https://github.com/mrakotosaon/pointcleannet`

4.2.2 Intra-patch accuracy study

To put our assumption to the test, we focus on comparing the U-Net and PointCleanNet architectures on an outlier classification task. The goal of both networks is to output whether a point is an outlier or not. We start by computing the accuracies of the two models we seek to compare, U-Net and PointCleanNet. The results are displayed in Fig. 3. The main remarks we can get from this plot are the following. It is clear that the PointCleanNet architecture is superior to the U-net one in the case of outlier detection. *The accuracy of the U-Net model goes down as we go further away from the center.* This is mainly because PointCleanNet has stable access to information. Since for each point, PointCleanNet receives a new patch centered at said point. On the other hand, the U-Net method takes as input a whole patch and directly outputs a prediction for each point in the patch. Fig. 3 seems to show then that there is loss of information at the edges of the patch. For PointCleanNet we display the global accuracy, since its accuracy is not affected by its distance to a given point, by construction.

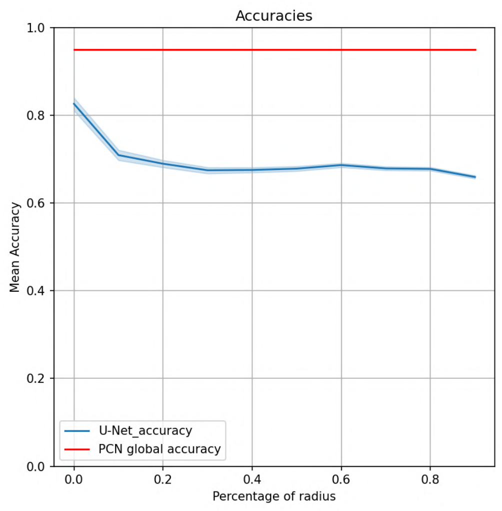


Figure 3: Mean accuracy plotted against the distance from the center of the patch studied. The blurry areas covering the lines represent the variance.

4.2.3 Smoothness study results

Algorithm 1 was then ran in the aforementioned environment, where the compared predictions are the probability vectors outputted by each network. The results are summarized in Fig. 4. We can see in this figure that the dissimilarity to the center behaves differently for both models. The U-net architecture displays a steady increasing trend. However, the plot shows a bimodal distribution for the PointCleanNet architecture. This justifies the statement we have proposed above: The U-net architecture's prediction space is of a smoother nature than the PointCleanNet one. It explains the gap we can see in Fig. 4 on the PointCleanNet KDE. Indeed a Kernel Density Estimate (KDE) is a statistical tool that helps to estimate the probability density function of a random variable. One can think of it as a continuous and smooth histogram. Thus, we learn that in the perspective of the PointCleanNet architecture changes in the "outlierness" of a point are abrupt. On the contrary, the U-net architecture seems to see points close together as having a similar "outlierness".

```
Consider $f$ an outlier detection network;
for a random point $x_0$ in a point cloud do
    Select the associated patch $X$;
    Apply $f$ to all points of the associated patch, call the results $P$ ;
    From $P$ extract $p_0$ the prediction of $x_0$ ;
    for $x$,$p$ the corresponding point,prediction pairs from $X$,$P$ do
        Store $\left(||x - x_0||_2, |p - p_0|\right)$
    end
end
```

Algorithm 1: Smoothness test

At a first glance this seems to show a weakness in the U-net architecture. A perfect outlier detector should be completely binary. A point is either

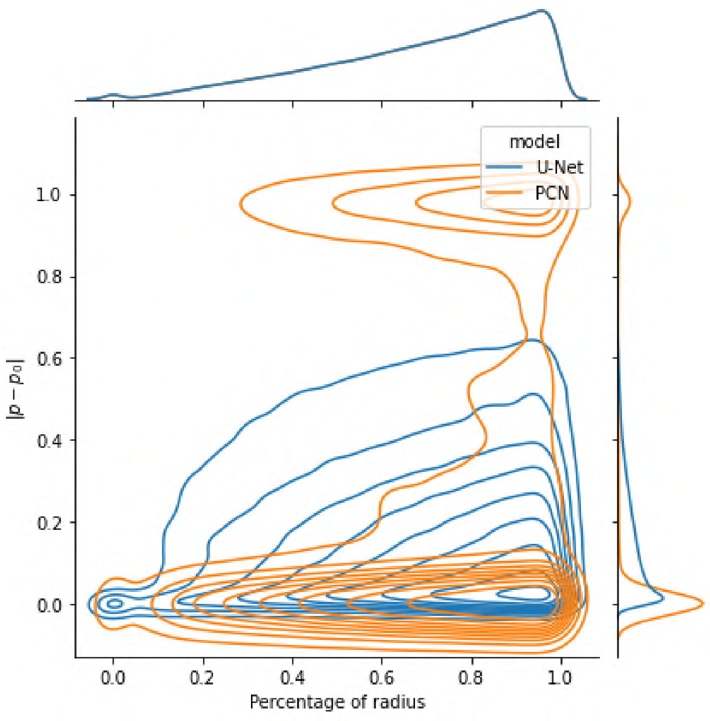


Figure 4: Kernel density estimate (KDE) plot. Compares distances in the input space and in the prediction space induced by different models. Here the x-axis was normalized by the patch radius. Note there are no values in the y-axis under 0 or over 1, the resulting overflow is made for visualization purposes.

in the underlying surface or it is not. We explore this by re-plotting the KDE plot using the "real" predictions of the networks. That is, instead of considering the differences between the probability vectors returned by the network, we apply a threshold to study the differences in the actual predicted labels. That is, if the probability of being an outlier is over 0.5 then it is an outlier, otherwise it is an inlier. In Fig. 5, the threshold has been applied. We can see that despite the continuous nature of the U-Net predictions both models manage to match the ground truth distribution quite well, the PCN one having a tighter fit in accordance to our previous section. Therefore,

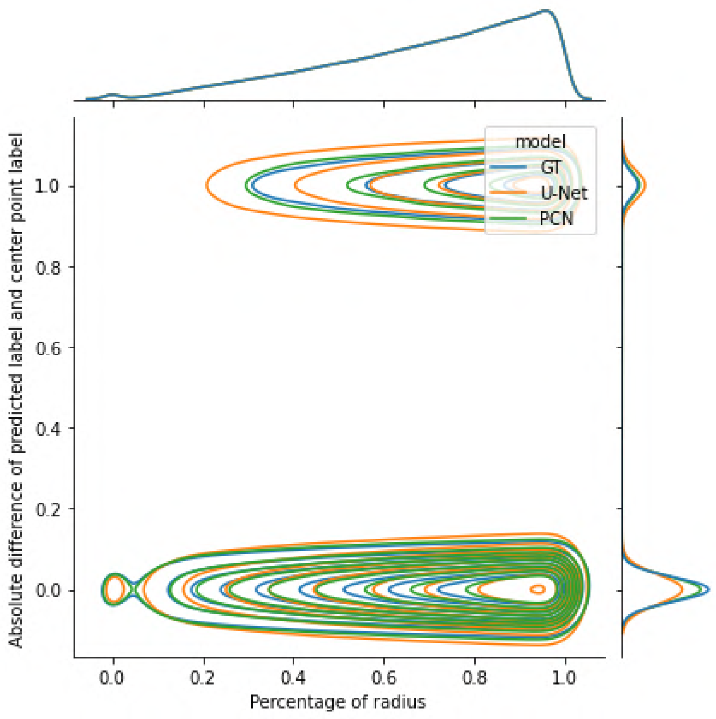


Figure 5: Kernel density estimate (KDE). Using the same notation as in Fig. 4. Using threshold for probability outputs.

showing that the continuity property of the U-Net architecture does not necessarily imply a lack of accuracy.

To conclude on these results, we have indeed shown that the U-Net architecture is of a smoother nature than the PointCleanNet one, on a binary semantic segmentation task. However, given the performance of each model, we have also discovered that a non-smooth model might be a better choice for the outlier removal task. Indeed, as we have briefly stated the "outlierness" scalar field is discontinuous by nature. Thus, this result does not come as a surprise. However, one must say that the U-Net architecture still managed to perform adequately. Therefore, it is still a viable option to explore, and to make predictions at all points in a given patch, which is more efficient.

5 U-Net architecture for Offset Regression

Let us now dive into the main contribution that this report proposes. We propose a cloud denoising architecture that solves the problems highlighted in section 1.1. We place ourselves in a version of the problem formulated in the ScoreDenoise paper [12]. That is we assume that the noisy clouds are constructed in the following way,

$$\mathbb{P}_{\mathcal{S}}^{N} = \{\mathbf{p} + \mathbf{n} \mid \mathbf{p} \in \mathcal{S}, \mathbf{n} \sim \mathcal{N}\}$$

see 1.1 for notation.

5.1 Network architecture

The architecture that we have chosen to solve the above problems is based on the concepts of the U-net architecture (section 3.1) and KPconv for our backbone [20].

5.1.1 Experimental Setup

We have trained using the PointCleanNet dataset[5]. This dataset is composed of 28 different shapes, which are split into 18 training shapes and 10 validation shapes. Furthermore we have a separate test set composed of 31 shapes [15].

An instance of our dataset is then a patch of any of the given shapes. These patches are composed of a maximum of 500 points. If the necessary points do not exist in the patch, we pad the rest, and make use of a mask to take into account only the real points during back-propagation. To train our network Gaussian noise is applied to all shapes. Since we have the underlying mesh information for every shape we can compute for each corrupted point the

[5]http://www.lix.polytechnique.fr/ mrakotos/pointcleannet.html

shortest vector that brings it to the surface. These vectors become then the target data. This is a process that is hugely time consuming, and therefore cannot be done online with the rest of the training. Therefore we store all this information and we use it repeatedly during training. In Fig. 6 we can see the shapes composing the PointCleanNet dataset.

5.1.2 Loss

We use the ground truth offset allowing to place the noisy points back to the underlying surface as our supervision. The offsets are computed using the euclidean shortest distance to the shape's 3D mesh. In section 5.2 is a brief study of an experiment that was conducted to compare different losses. The losses that we explored are the following.

L1 loss. The classical L1 loss. Consider $x, y \in \mathbb{R}^3$

$$L1(x, y) = \sum_{i=1}^{3} |x_i - y_i|.$$

The Chamfer Loss. Given that our network works in a patch to patch basis we can apply a patch loss. Consider a two non-empty patches X, Y. The chamfer distance is defined as

$$d_{CD}(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} ||x - y||_2^2 + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} ||x - y||_2^2.$$

The Chamfer Loss (L1). In this experiment we wanted to promote sparsity near the vectors with 0 values, therefore we implemented a Chamfer Loss that uses L1 as the distance. The chamfer L1 distance is given by,

$$d_{CD(l1)}(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} |x - y| + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} |x - y|.$$

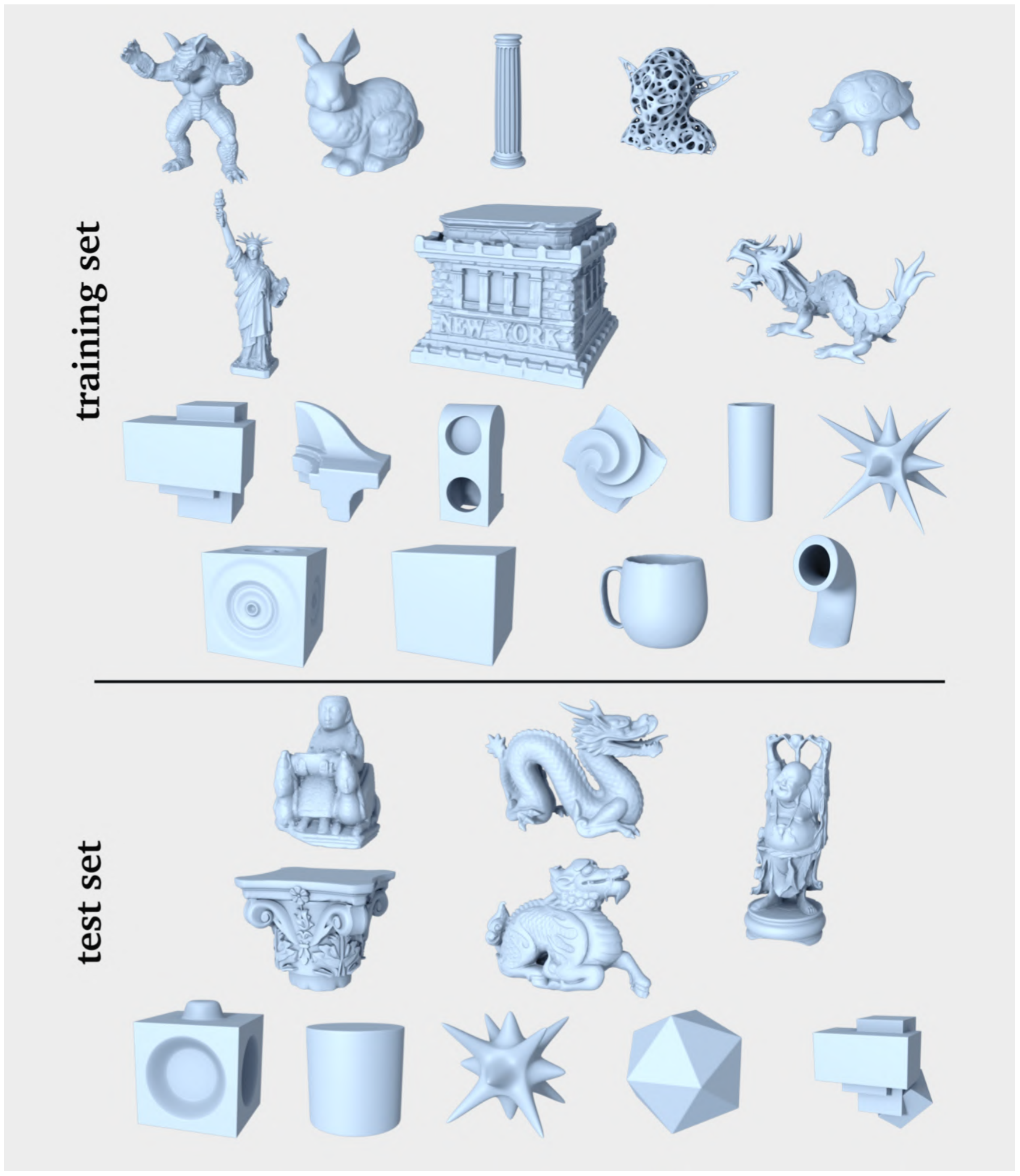


Figure 6: PointCleanNet shapes from the training and validation set. Figure extracted from `http://www.lix.polytechnique.fr/~mrakotos/pointcleannet.html`

5.1.3 Execution at test time

Since we train our network with only patches at training time, execution at test time is slightly different. We use a grid-sub sampling technique to chose a set of center points. Then the network is applied to each patch associated with the given center point. All of the information is then averaged per point, we chose the parameters in the grid sub-sampling method to make sure that all points will be seen at least once. Once all the predicted offset vectors have been averaged, we apply these ones to the noisy cloud to obtain the final predicted denoised cloud.

5.2 Performance of different losses

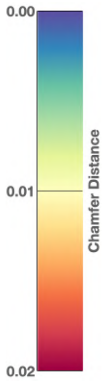
Now, let us present the results on the performance of the different losses that we have tested. Each experiment was ran using the exact same architecture, with the same hyper parameters, and training each network for 100 epochs. The training set was corrupted using Gaussian noise with a standard deviation (std) of 0.1%, 0.5% the shape size (respectively for each test), and with mean 0. The test was conducted on our testing dataset [15], with the same perturbation as the training set. The networks were ran on all test shapes yielding their respective denoised shapes. Finally, we computed the Chamfer Distance to the clean shapes. In the table below, the results are displayed. We call the Chamfer loss ratio,

$$\text{Chamfer Ratio} = \frac{\text{Chamfer Distance between the denoised and clean shape}}{\text{Chamfer Distance between the noisy and clean shape}}$$

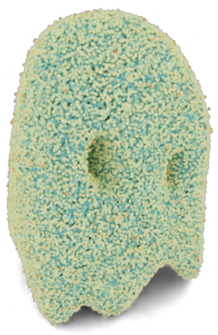
Chamfer Loss ratio		
Model Name	0.5 %	0.1%
Noisy	1	1
L1	**0.20**	1.45
Chamfer	0.51	**0.97**
Chamfer + L1	0.25	1.04
Chamfer(L1)	1.6	54.14
Chamfer(L1) + L1	0.39	19.44

The table above shows that quantitatively the best loss to train our Network on is the L1 loss. It manages to decrease the Chamfer Error by 80% on the std 0.5% case. However, it can be seen that no loss managed to denoise in a significant way the cloud with noise of std 0.1%. This might indicate that for the U-Net architecture, such little noise is too hard to notice. Some these networks actually "increased" the noise of the shapes. After inspection what seems to happen is that in some cases the network does not "learn anything" and outputs a constant vector. This leads to an affine translation of all points, which of course increases the overall noise ratio.

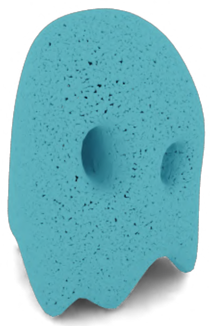
Let us now dive into some qualitative results, see Fig. 7. We observed empirically that, despite poor performance on the quantitative test, some losses induced clouds that seemed to be clean. As we have briefly mentioned, we noticed that some models where shifting the clouds. Nevertheless, in some cases despite this shifting some networks actually performed some denoising. Note that this cannot be said of all poorly performant networks, specially on the 0.1% case. Thus, in the qualitative test below this shift was corrected by applying the Iterative Closest Point (ICP) algorithm. This algorithm allows to minimize the difference between two clouds of points, by using rigid deformations (i.e rotation and translation). One can see then in Fig. 7 that the results are quite surprising. It appears that the worst performing losses



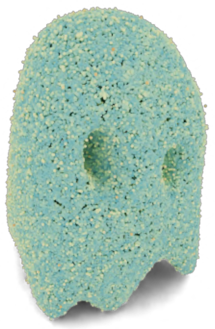
(a) Color Scale



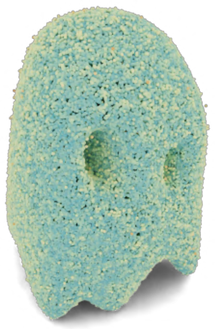
(b) Noisy Shape



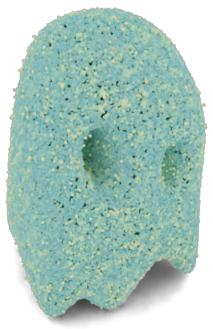
(c) Clean Shape



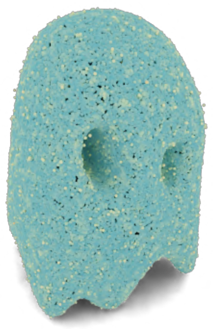
(d) Chamfer



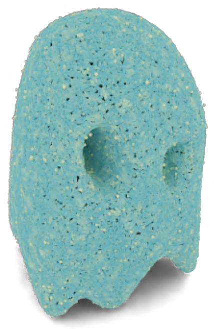
(e) Chamfer + L1



(f) Chamfer(L1)



(g) Chamfer(L1) + L1



(h) L1

Figure 7: Qualitative tests

return in fact descent outputs (0.5 % case), specially the modified Chamfer Distance (L1). This is very intriguing and calls for further investigation. However, in this report we are specifically interested in developing a tool with the least amount of post-processing. Therefore we will move on based on the results seen in the above table, and pick the L1 loss as our main loss for the following experiments.

5.3 Comparison with PointCleanNet

We compare now our best U-Net architecture against the PointCleanNet architecture results on the denoising problem. In the table below, are displayed the different performances. The U-Net model was trained using the L1 loss, and the same shapes as before, but now with gaussian noises of std $0\%, 0.5\%, 1\%, 1.5\%, 2.5\%$ of the original shape. This setting matches the training of PointCleanNet [16]. Note that these different gaussian noises were applied point wise, so the points of a given noisy shape do not follow a specific gaussian distribution. One could argue that they follow a gaussian mixture distribution. The PointCleanNet architecture was tested using the available pretrained weights [6].

Chamfer Loss ratio		
Model Name	0.5 %	0.1%
U-Net	0.22	1.3
PointCleanNet Pretrained	0.32	1.01

From one can see in the above table, the U-Net outperforms the PointCleanNet on the test set for noisy shapes with gaussian noise of std 0.5%. However, it seems that in the case of gaussian noise with std 0.1% both networks could not do anything. Moreover, the U-Net architecture actually deformed the noisy clouds even more (refer back to the shifting phenomenon when "nothing is learnt"). In Fig. 8 we can see a case where the U-Net architecture outperforms the PCN one. One can see that the error, represented by the colors of the shapes, is significantly lower in the case of the

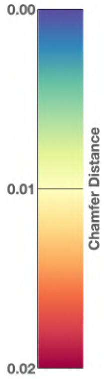
[6]http://www.lix.polytechnique.fr/ mrakotos/pointcleannet.html

U-Net architecture. We also present one of the worst performances, relatively to PointCleanNet, of U-Net in the test set. This is shown in the appendix Fig. 12. Here we can see that the opposite is true, where the color of the PointCleanNet shape is much lighter. However, if one looks carefully, one can see that the U-Net shape seems to be “cleaner” than the PointCleanNet one. Indeed, the PointCleanNet one seems to have more outlier points near the edges. This could be an indication that the U-Net model is prone to a slight over smoothing effect.

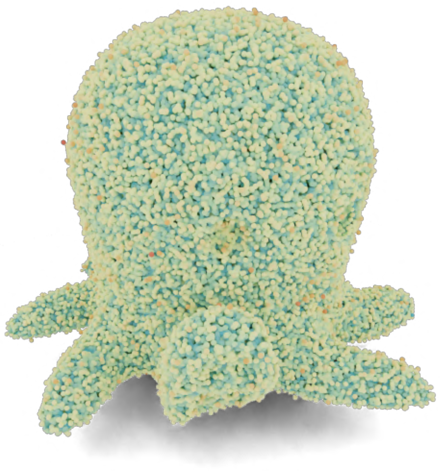
6 Addition of a discriminator component

As we have hinted in the beginning of this report, the integration of a discriminator component to our main pipeline could be greatly beneficial. We got the inspiration from the idea of GAN’s. Mainly the use of them on this paper [8], where the authors use a GAN structure to complete missing holes in images. One could think about their approach as inputting a corrupted object and outputting a ”clean” one. In the same way in one could attempt to improve the offset regression architecture by training it paired to a discriminator component that tells if a patch is clean or noisy. Unfortunately, training GANs is usually not a straight forward task [21]. The author of this report has started experimenting with this idea, but has not yet been completely successful. In Fig. 9 is displayed a proposed architecture.

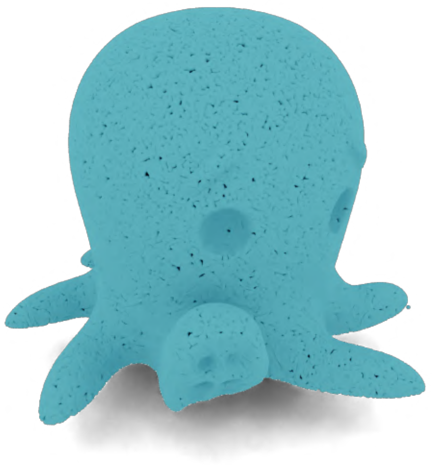
In the appendix (see Figs. 13,14) we can see figures of some first results. One can note that the qualitative tests that have been obtained with the models that were trained with the GAN pipeline seem to be sharper near the underlying surface. However, these examples also have many more outliers. After some inspection one can deduce that some drift phenomenon is taking place. The hypothesis is that the offset regression architecture is managing to “fool” the discriminator by somehow “getting rid” of the noisy points, and



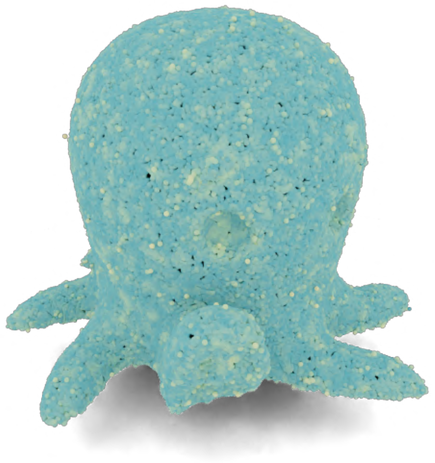
(a) Color Scale



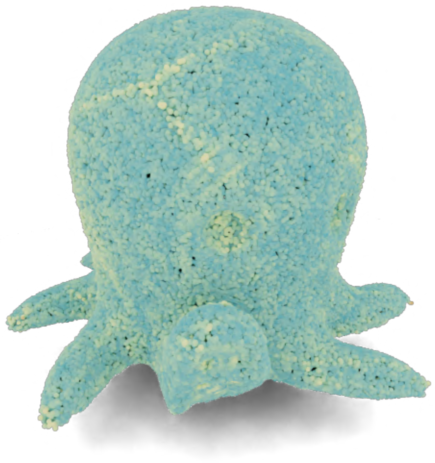
(b) Noisy Shape



(c) Clean Shape



(d) Denoised with U-Net



(e) Denoised with PCN

Figure 8: Qualitative tests

cleaning up the points closer to the true surface.

It would be thus interesting to adapt our network to also perform outlier detection. Doing this is relatively straight forward, since the only structural change required is the addition of a 4-th coordinate in the prediction vector that indicates the probability of a point being an outlier. This implementation could perhaps lead to even better results.

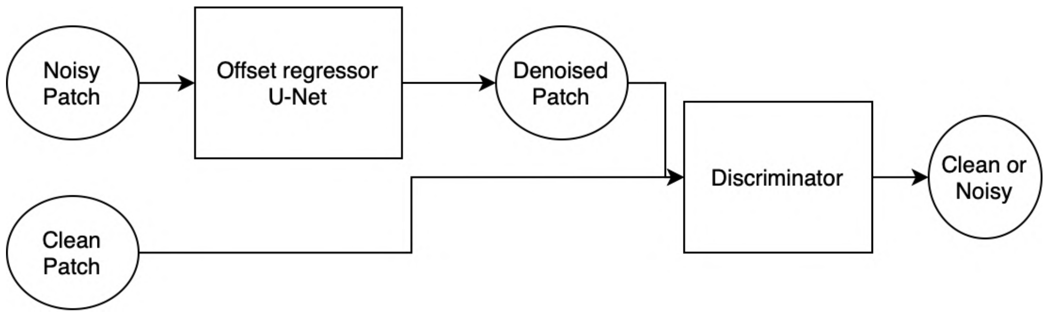


Figure 9: Addition of Discriminator Component to our pipeline

7 Conclusion

To conclude, in this report we have re-purposed a semantic segmentation architecture for the offset regression task. This allowed us to solve the raised problems in section 2.2. Our U-Net based architecture,

- Does not require iterative cleaning
- Uses a meaningful loss
- Uses a patch prediction approach

It is quite astounding that an “off the shelf” semantic segmentation architecture gives the presented results. And with these few changes to the preexisting structure we have managed to beat a well established point denoising technique, PointCleanNet. Nonetheless, it is clear from our qualitative examples that the final denoised shapes could still use some improvement. The author of this report does not believe that we have yet exploited the U-Net architecture to its fullest, and that further investigation could lead to even better results.

8 Future Work

The U-Net architecture is still full of potential and with more work could lead to even more interesting results. A first step involves the re exploration of the Chamfer(L1) loss. As we saw in section 5.2, the performance of this loss was underwhelming. However, after the ICP post-processing step, the qualitative examples showed promising results. It would be thus interesting to try to add some regularization to the training pipeline to drive the model coupled with the Chamfer(L1) loss to be able to obtain the promising results without the use of any post-processing.

Moreover, the hyper-parameters of the network were not explored in this report. Perhaps changing the number of points per patch, or the patch radius could have an impact on performance. It would also be of interest to perform different kind of studies with the current network while changing the hyper-parameters or the test set. For instance, plot the performance of the network with respect to different kinds of noise.

Finally, this network could be easily adapted to classify outliers and perform offset regression at the same time. For instance, by adding a 4-th coordinate to the output that represents the probability that a given point is an outlier. This could potentially make the network more robust, and as we have mentioned could benefit while training in the GAN pipeline.

9 Appendix

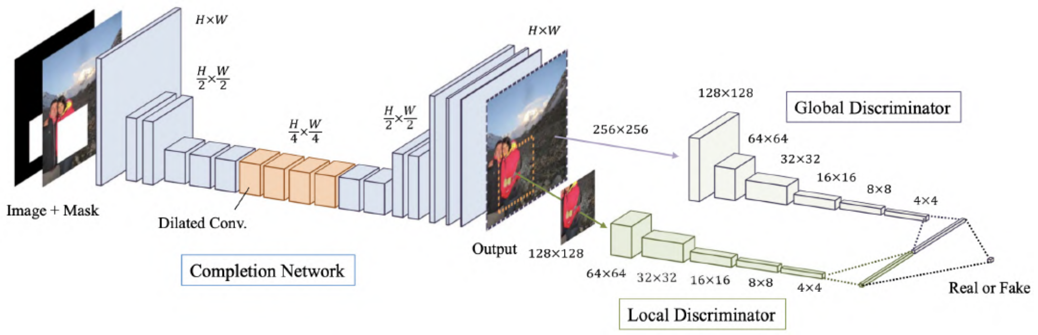


Figure 10: Example of GAN used for a "cleaning" problem. Figure extracted from [8].

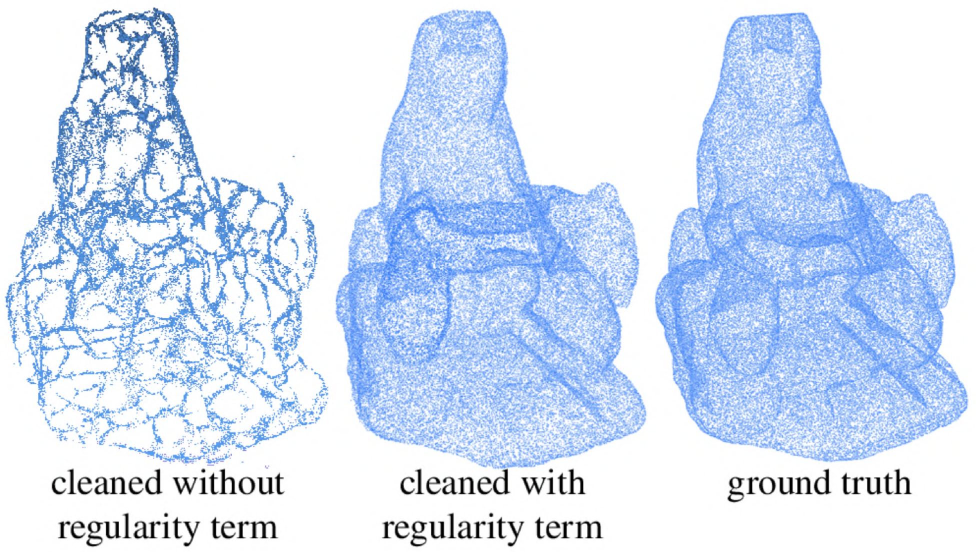
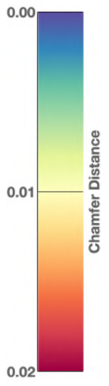
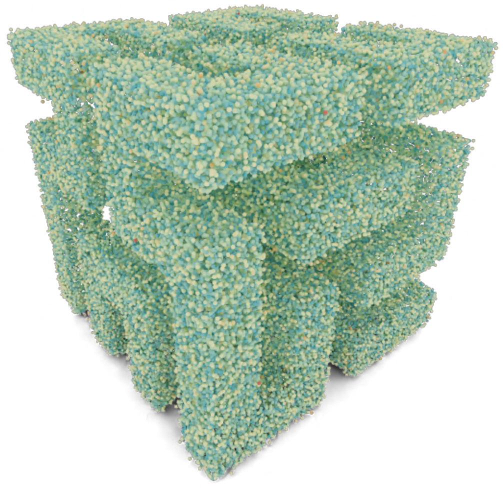


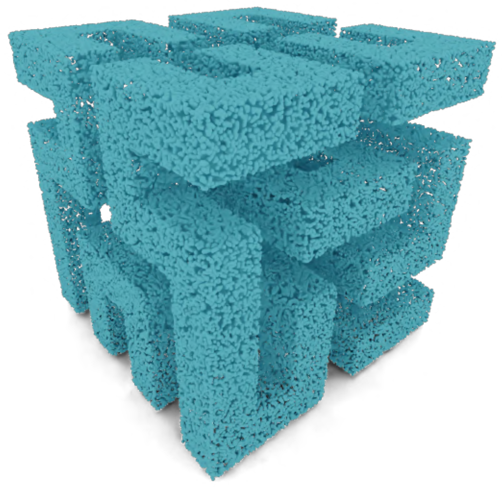
Figure 11: Display of filament structures when using PCN. Figure extracted from [16]. The regularization terms involves minimizing the squared distance of the prediction to the farthest point in the local patch.



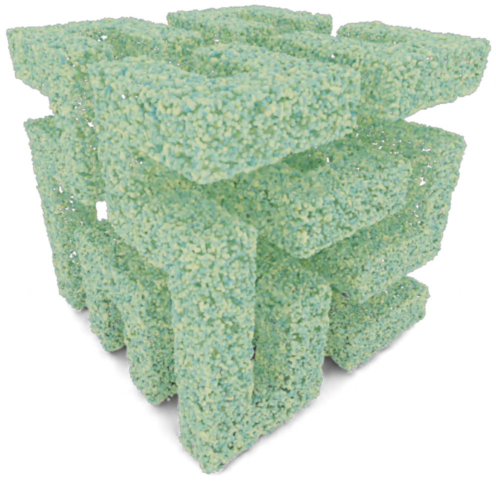
(a) Color Scale



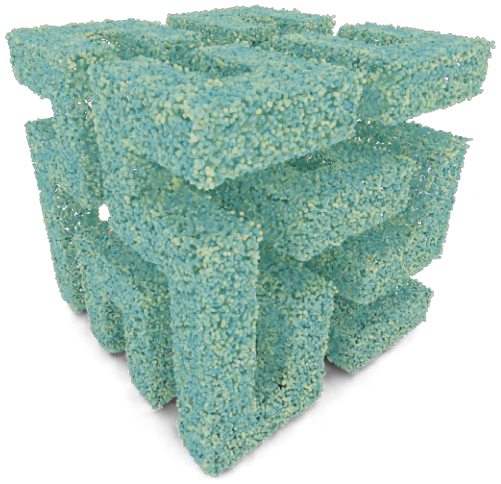
(b) Noisy Shape



(c) Clean Shape



(d) Denoised with U-Net



(e) Denoised with PCN

Figure 12: Qualitative tests

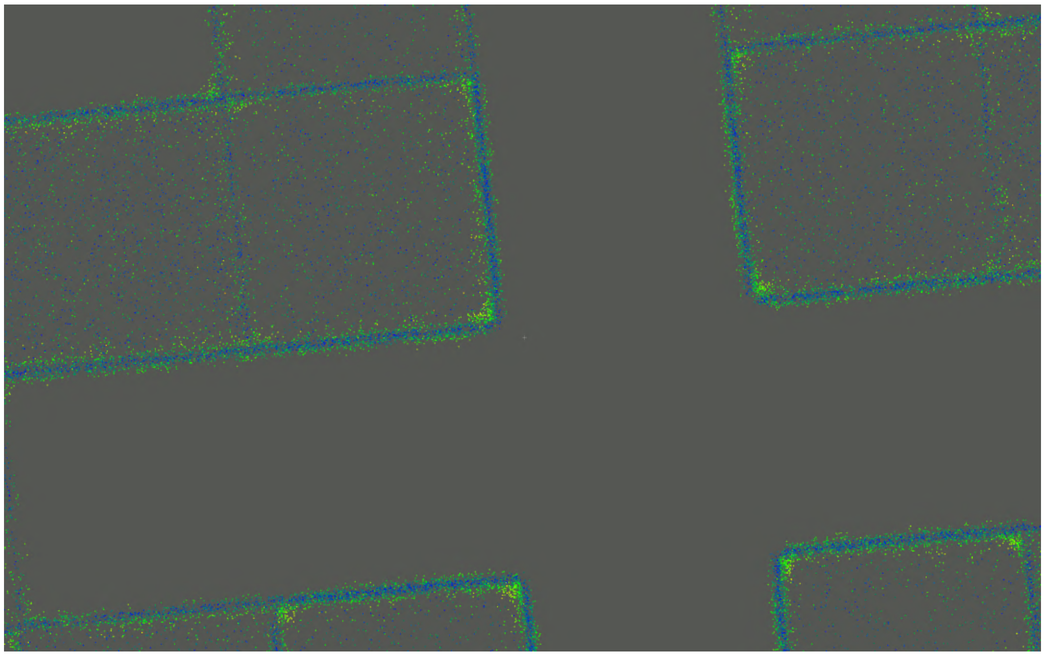


Figure 13: Qualitative test with no GAN training.

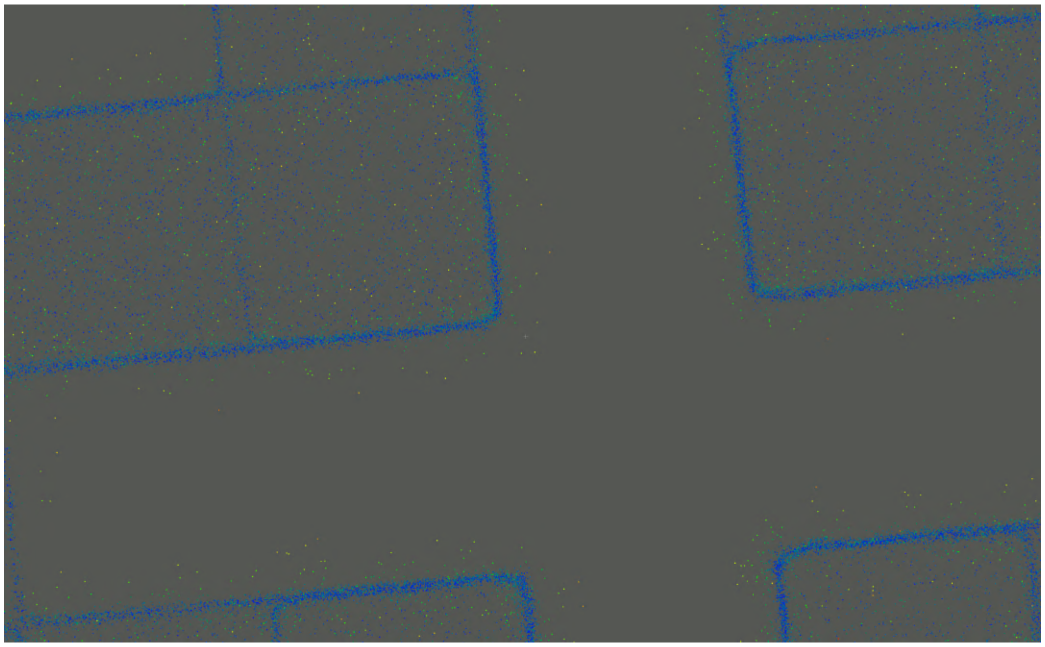


Figure 14: Qualitative test with 100 epochs of GAN training.

References

[1] Zailiang Chen, Ziyang Zeng, Hailan Shen, Xianxian Zheng, Peishan Dai, and Pingbo Ouyang. Dn-gan: Denoising generative adversarial networks for speckle noise reduction in optical coherence tomography images. *Biomedical Signal Processing and Control*, 55:101632, 2020.

[2] Ulrich Clarenz, Martin Rumpf, and Alexandru Telea. *Fairing of point based surfaces*. IEEE, 2004.

[3] Shachar Fleishman, Daniel Cohen-Or, and Cláudio T Silva. Robust moving least-squares fitting with sharp features. *ACM transactions on graphics (TOG)*, 24(3):544–552, 2005.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[5] Paul Guerrero, Yanir Kleiman, Maks Ovsjanikov, and Niloy J Mitra. Pcpnet learning local shape properties from raw point clouds. In *Computer Graphics Forum*, volume 37, pages 75–85. Wiley Online Library, 2018.

[6] Xian-Feng Han, Jesse S Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, and Liping Xiao. A review of algorithms for filtering the 3d point cloud. *Signal Processing: Image Communication*, 57:103–112, 2017.

[7] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. Total denoising: Unsupervised learning of 3d point cloud cleaning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 52–60, 2019.

[8] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG)*, 36(4):1–14, 2017.

[9] Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan. *arXiv preprint arXiv:1810.05795*, 2018.

[10] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: A point cloud upsampling adversarial network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[11] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A closer look at local aggregation operators in point cloud analysis. In *European Conference on Computer Vision*, pages 326–342. Springer, 2020.

[12] Shitong Luo and Wei Hu. Score-based point cloud denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4583–4592, 2021.

[13] Francesca Pistilli, Giulia Fracastoro, Diego Valsesia, and Enrico Magli. Learning graph-convolutional representations for point cloud denoising. In *European conference on computer vision*, pages 103–118. Springer, 2020.

[14] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[15] Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy J Mitra, and Maks Ovsjanikov. Learning delaunay surface elements for mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22–31, 2021.

[16] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J Mitra, and Maks Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. *Computer Graphics Forum*, 2019.

[17] José Carlos Rangel, Vicente Morell, Miguel Cazorla, Sergio Orts-Escolano, and José García-Rodríguez. Object recognition in noisy rgb-d data. In *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pages 261–270. Springer, 2015.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[19] Oliver Schall, Alexander Belyaev, and H-P Seidel. Robust filtering of noisy scattered point data. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 71–144. IEEE, 2005.

[20] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.

[21] Weiyao Wang, Du Tran, and Matt Feiszli. What makes training multimodal classification networks hard? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12695–12705, 2020.

[22] Xin Wang, Hui Guo, Shu Hu, Ming-Ching Chang, and Siwei Lyu. Gan-generated faces detection: A survey and new perspectives. *arXiv preprint arXiv:2202.07145*, 2022.

[23] Lang Zhou, Guoxing Sun, Yong Li, Weiqing Li, and Zhiyong Su. Point cloud denoising review: from classical to deep learning-based approaches. *Graphical Models*, 121:101140, 2022.

[24] Huang Zuowei, Huang Yuanjiang, and Huang Jie. A method for noise removal of lidar point clouds. In *2013 Third International Conference on Intelligent System Design and Engineering Applications*, pages 104–107. IEEE, 2013.